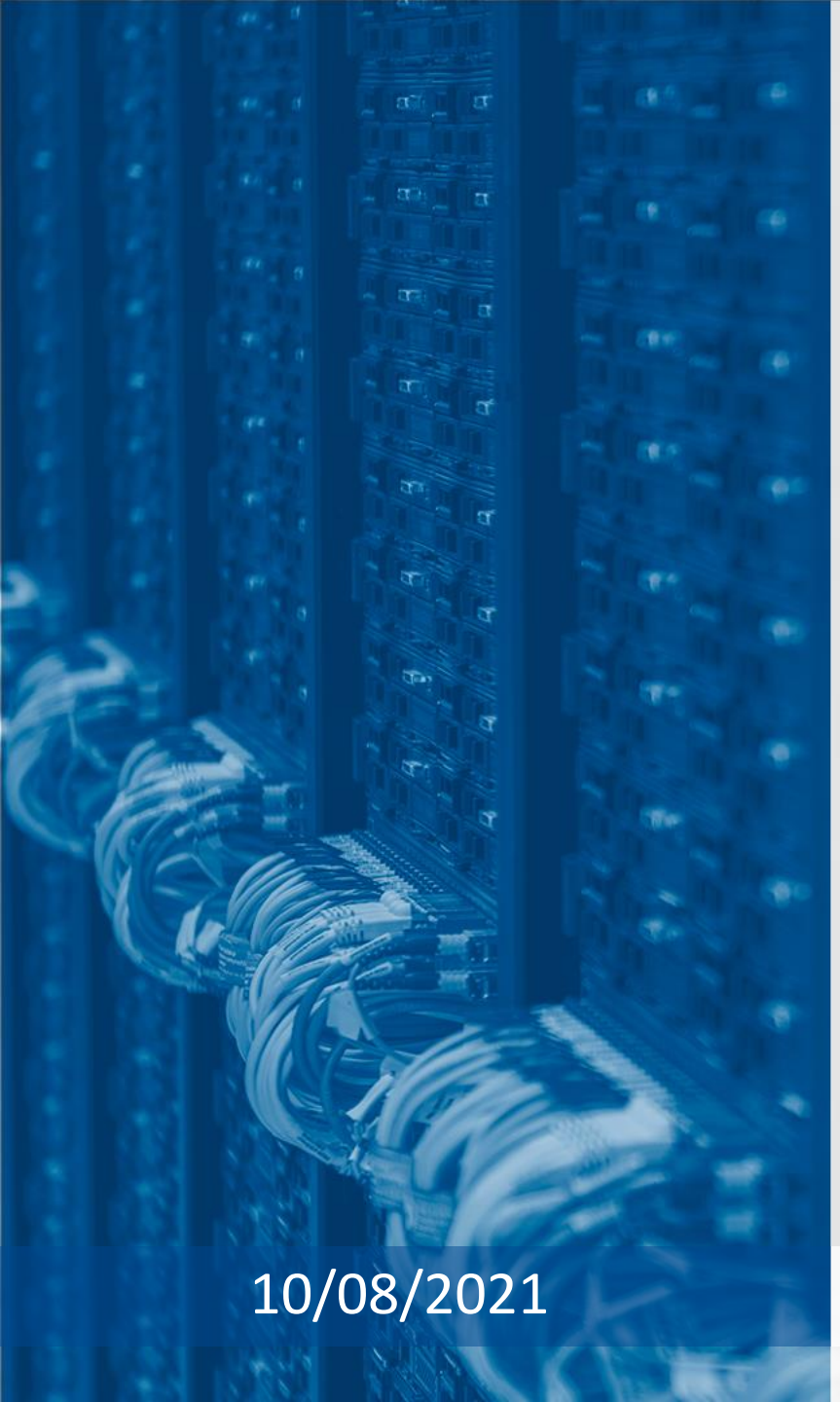


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Combining Dynamic Concurrency Throttling with Voltage and Frequency Scaling on Task-based Programming Models

Antoni Navarro <antoni.navarro@bsc.es>
Arthur F. Lorenzon <aflorenzon@unipampa.edu.br>
Eduard Ayguadé <eduard.ayguade@bsc.es>
Vicenç Beltran <vbeltran@bsc.es>

10/08/2021

International Conference on Parallel Processing (Virtual)

Contents

- Motivation
- Objectives
- Monitoring and Prediction Infrastructure
- Integration of DCT/DVFS Techniques
- Performance Evaluation
- Conclusions and Future Work

Motivation

- With each passing day, current systems get closer to Exascale performance
- For some years already, power-performance has become a key metric in the HPC community
- Improving the energy consumption of applications while also improving performance (or leaving it unhindered) is still being studied to this day

Motivation

- Thus, to improve power consumption, works often adopt strategies based on:
  - Dynamic Concurrency Throttling (DCT)
    - Adapt the number of computing resources at runtime
  - Dynamic Voltage and Frequency Scaling (DVFS)
    - Tweak the frequency at which resources operate
  - A combination of both
    - Not straightforward, hard to combine, depends on workload

	Compute-bound	Memory-bound	Balanced
DCT		X	X*
DVFS		X	X*
DVFS (uncore)	X		X*

Motivation

- Task-based parallel programming models, such as OmpSs-2, have emerged as an alternative to fully yield the potential of these systems

- Currently, OmpSs-2 has a monitoring infrastructure that predicts the elapsed time of tasks for better scheduling opportunities

- Could we extend such an infrastructure with more metrics and features, so that the runtime would be able to infer the computational profile of tasks, which in turn would improve the scheduling opportunities?

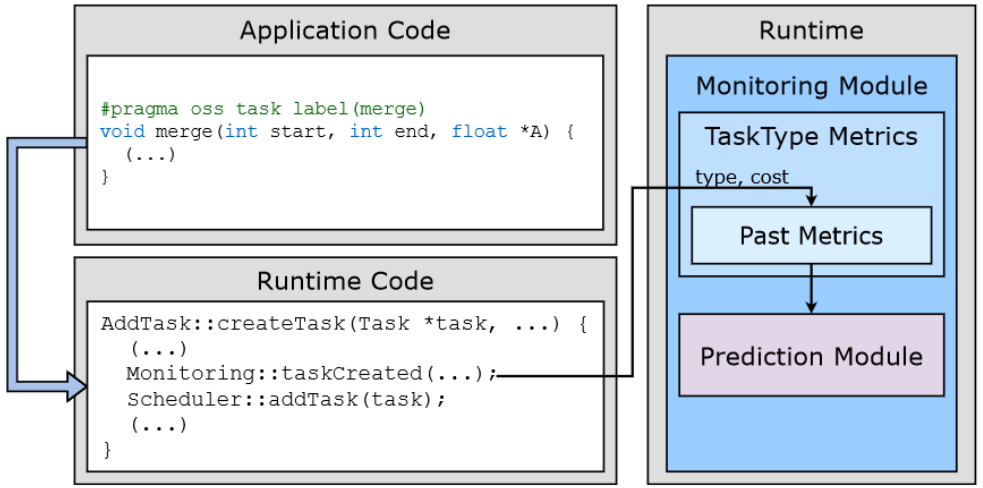


Figure : Monitoring and Prediction Infrastructure.

Objectives

- To combine DCT and DVFS strategies properly, we propose:

1. An infrastructure that precisely categorizes workloads based on their computational profile

2. Performing the categorization in an on-line manner, at runtime, and with a negligible overhead in execution time

3. Based on the categorization of workloads, create heuristics that properly employ a combination of DCT and DVFS for both core and uncore units, automatically

Monitoring and Prediction Infrastructure

- To correctly categorize workloads (tasks), we need to record and predict:
  1. The limits in Mem. BW of the underlying system
  2. Their effective memory bandwidth
     - For this, we need their elapsed time and the number of LLC misses

- With the previous information we can infer:
  1. The effective memory bandwidth per core (if we distribute bandwidth equally)
  2. The "intensiveness" in mem. BW of a task

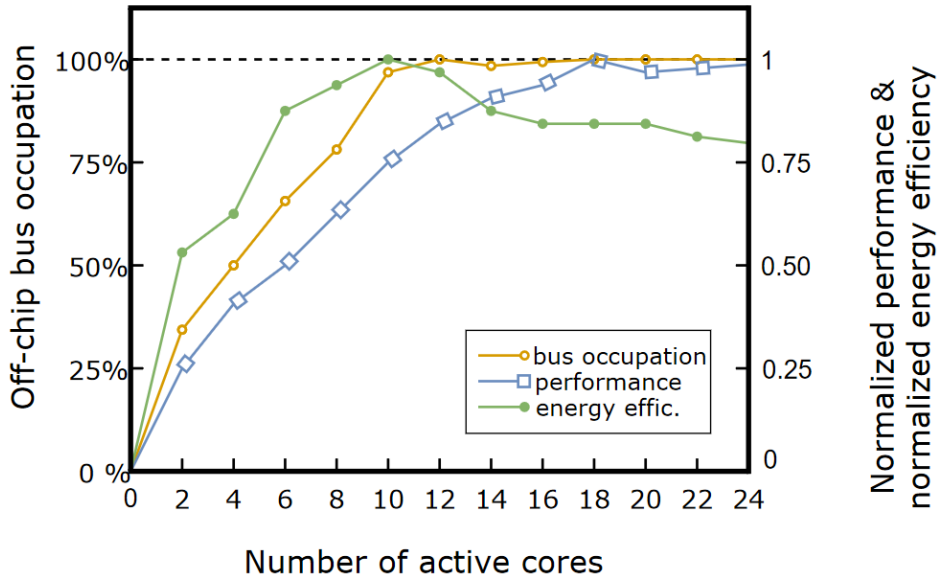


Figure 1: Off-chip bus saturation VS. performance.

Integration of DCT and DVFS Techniques

- Determining the “intensiveness” in mem. BW of a workload consists in adding the individual metrics of each executing task
- If the overall workload is highly compute-bounded, we use DVFS in uncore units to limit the frequency of the memory controllers, reducing the power consumption
- If the overall workload is highly memory-bounded, we either use DCT to limit the number of available threads, or DVFS to tune down the frequency of active cores
- Lastly, if the workload is balanced, we use either DCT or uncore DVFS depending on the number of tasks and their individual intensiveness

Evaluation – Overhead and Accuracy

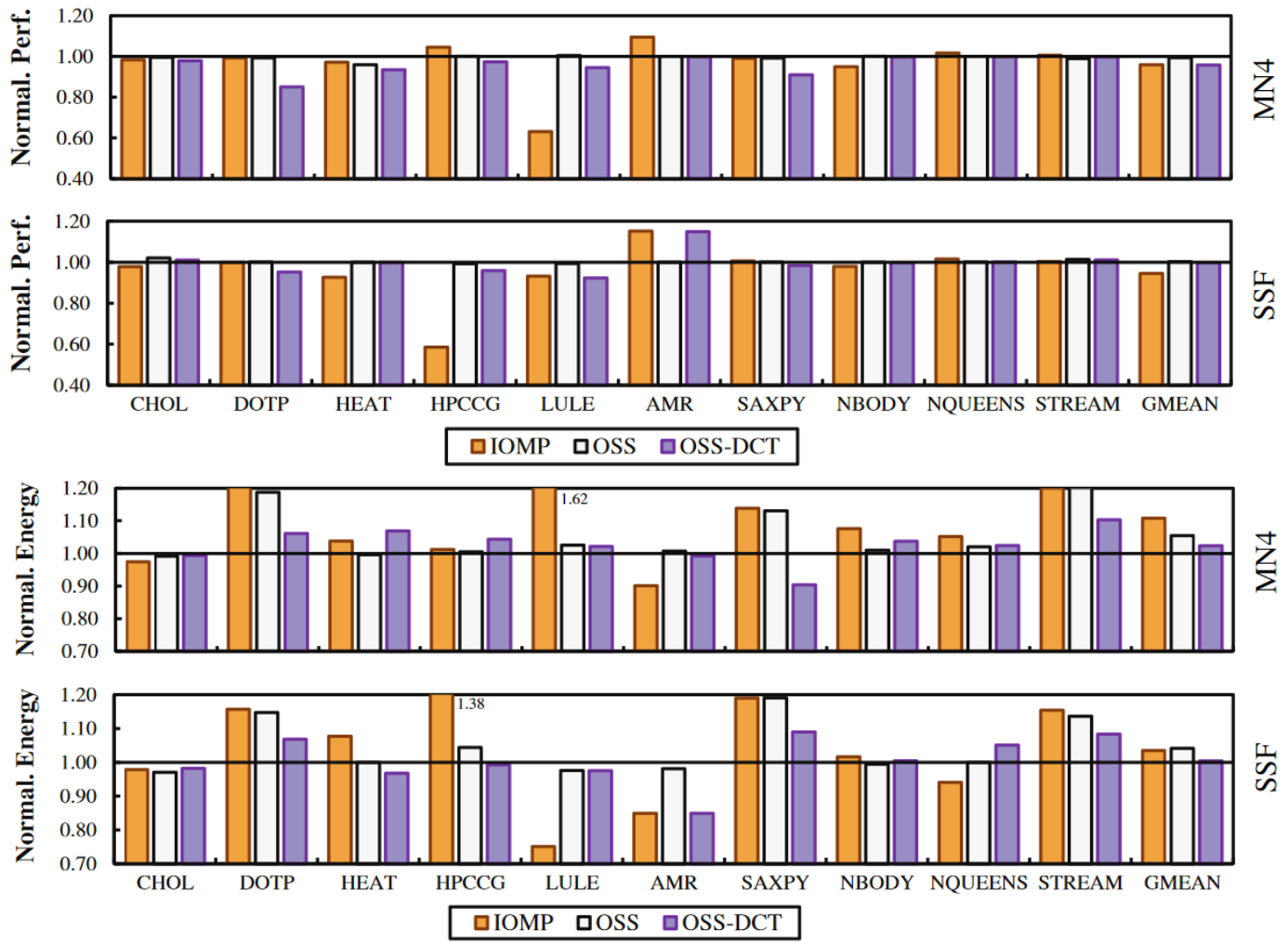
- Overhead – Negligible overall
  - Worst case scenario < 5% in an application with lots of fine-grained task
- Accuracy – High precision when necessary
  - Worst case scenario in hardware counter predictions for some compute-bounded workloads (variability jeopardizes accuracy)

		Cholesky	Dotproduct	Heat	HPCCG	LULESH	MiniAMR	Saxpy	NBody	Stream
MN4	*overhead*	0.17%	1.00%	3.29%	0.72%	3.36%	0.23%	2.90%	0.16%	0.09%
	acc. time	93.83%	87.91%	91.82%	82.39%	86.70%	91.39%	94.75%	99.82%	78.69%
	acc. hwc	88.97%	99.77%	99.38%	82.24%	74.54%	87.64%	99.83%	48.06%	99.07%
SSF	*overhead*	1.99%	2.03%	2.29%	4.87%	3.17%	0.29%	0.20%	0.48%	0.07%
	acc. time	94.85%	82.13%	87.32%	77.64%	82.19%	88.23%	82.41%	91.02%	95.61%
	acc. hwc	93.01%	91.59%	91.27%	72.02%	71.57%	84.66%	94.55%	45.95%	94.05%

Evaluation – DCT

Higher = better

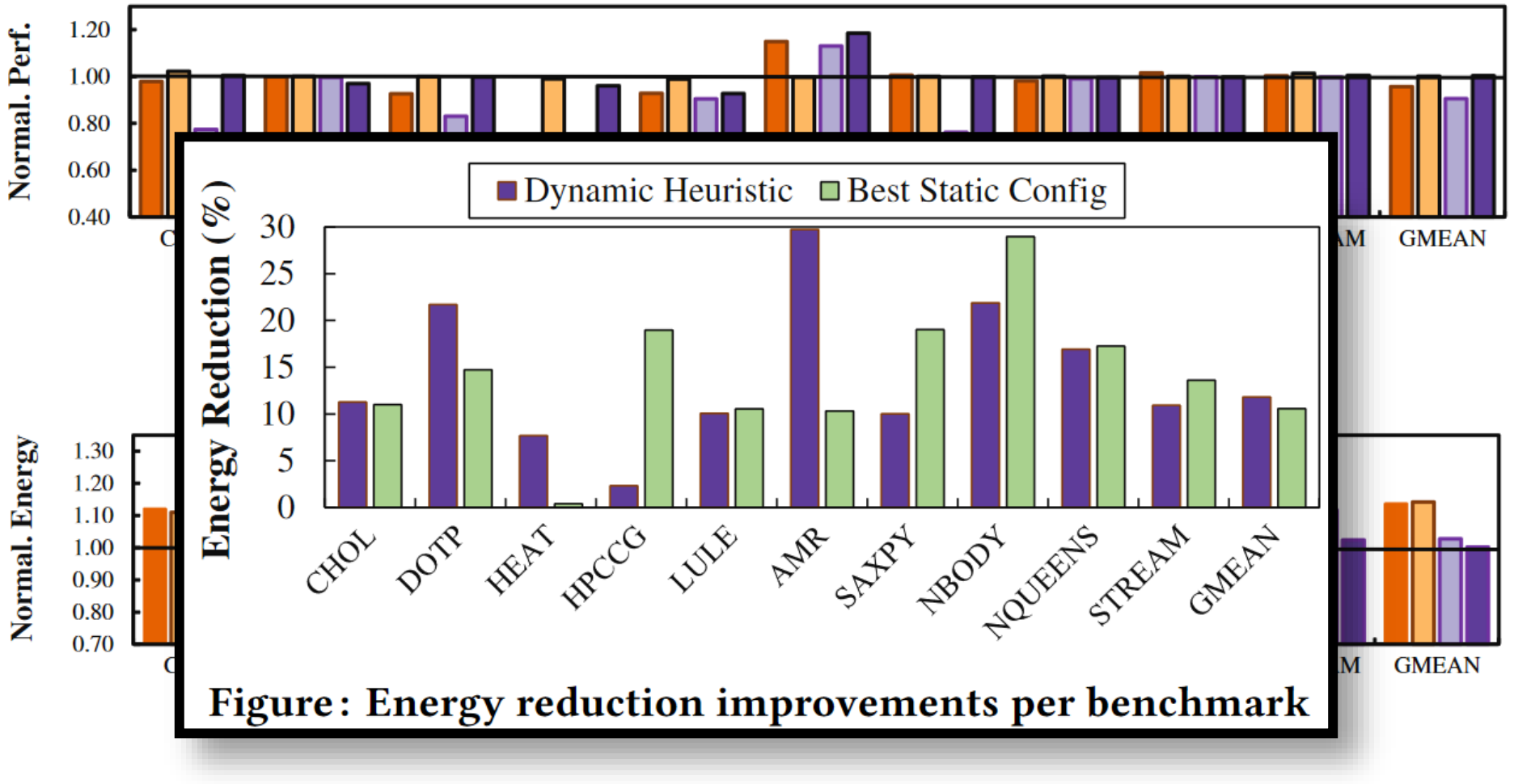
Lower = better



Evaluation – DVFS

Higher = better

Lower = better



Evaluation – In-Depth Analysis

- Execution traces of the activity of CPUs over time of miniAMR
  - Results show that while performance is unhindered, energy consumption is improved
  - Baseline Avg. CPU Utilization: 27.2
  - DVFS+DCT Avg. CPU Utilization: 24.4

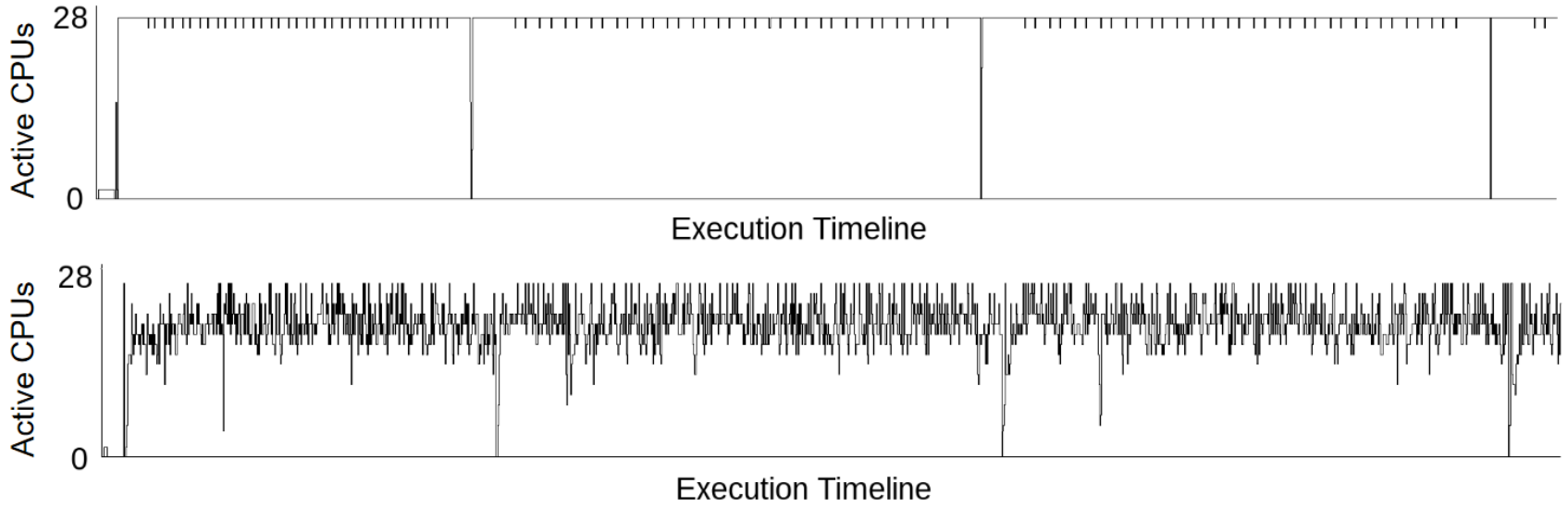
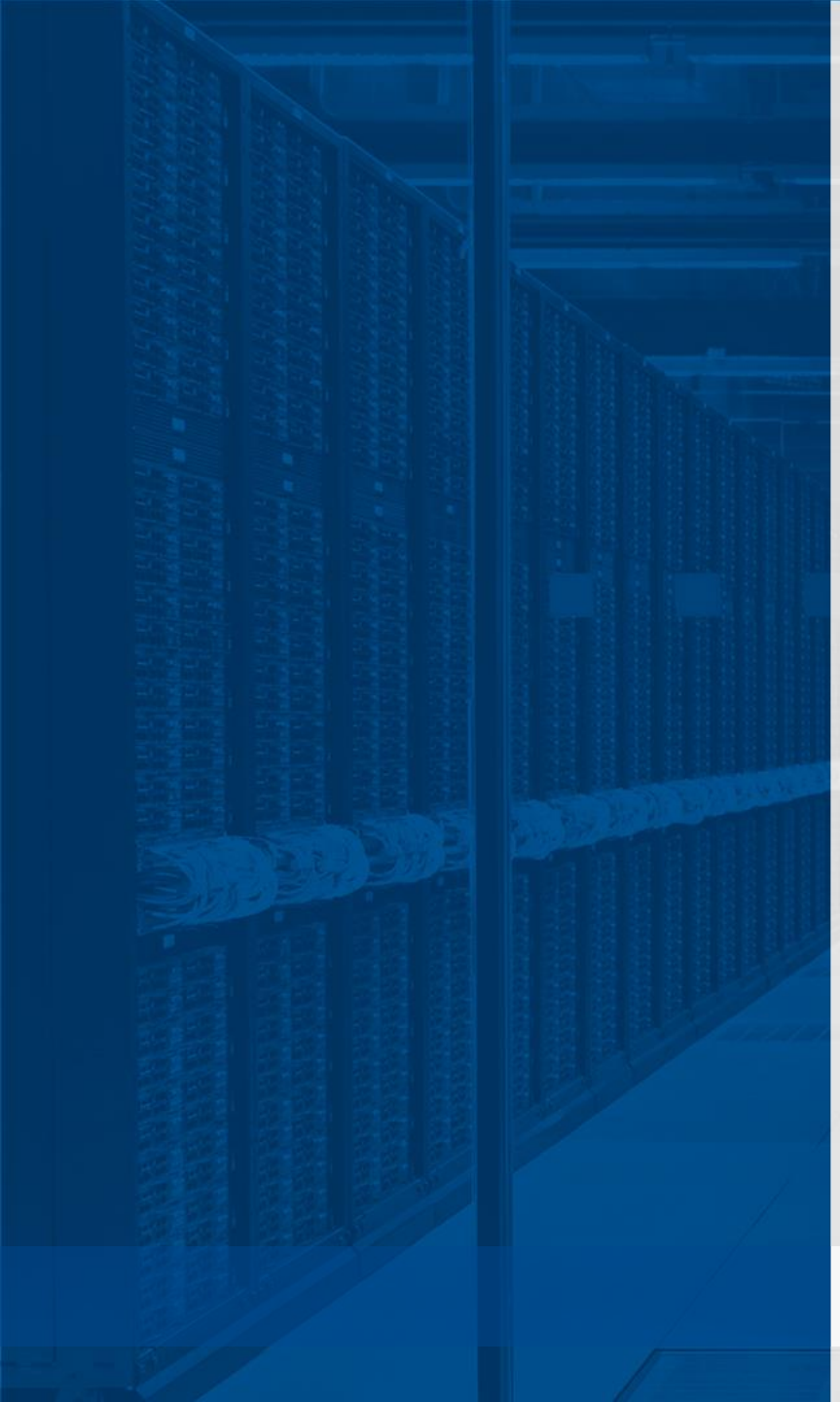


Figure: Active CPUs over time – Baseline (Top) vs. DVFS+DCT (Bottom)

Conclusions and Future Work

- Combining of DCT and DVFS is not straightforward, however our analysis shows that it can provide up to 15% better energy efficiency in average, regardless of system or workloads
- DVFS uncore combined with core DCT yields the most benefits out of all the studied combinations
- Employing techniques such as DVFS in other resources like accelerators could benefit heterogeneous systems
- Furthermore, hybrid applications could benefit from a system-wide scheduling technique that takes workload categorization and DCT/DVFS techniques into account



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Combining Dynamic Concurrency Throttling with Voltage and Frequency Scaling on Task-based Programming Models

Antoni Navarro - antoni.navarro@bsc.es